REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	10 Jan 1997	Final

4. TITLE AND SUBTITLE
Genetic Algorithms and Evolutionary Programming

5. FUNDING NUMBERS
N61339-96-D-0002

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lockheed Martin Corporation
Information Systems Company
12506 Lake Underhill Road
Orlando, FL 32825

8. PERFORMING ORGANIZATION REPORT NUMBER
ADST-II-CDRL-023R-9600239A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32826-3224

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
A - Approved for public release; distribution unlimited.

13. ABSTRACT *(Maximum 200 Words)*

This research activity examines the current state-of-the-art in modeling the command decision process and implementing such models in software. The primary and initial target application is in automated command agents for DIS/ADS. This report was prepared for the Command Decision Modeling ADST II Delivery Order in accordance with the following documents:

a) Command Decision Modeling Overview (ADST-II-CDRL-023A-9600236)
b) Functional Description of a Command Agent (ADST-II-CDRL-023A-9600237)
c) Rule Based Systems (ADST-II-CDRL-023A-9600238)
d) Genetic Algorithms and Evolutionary Programming (ADST-II-CDRL-023A-9600239)
e) Petri Nets and Colored Petri Nets (ADST-II-CDRL-023A-9600240)
f) Neural Networks and Bounded Neural Networks (ADST-II-CDRL-023A-9600241)
g) Case-Based Reasoning (ADST-II-CDRL-023A-9600242)

14. SUBJECT TERMS	15. NUMBER OF PAGES
Genetic Algorithms; Evolutionary Programming;GA;EP ADST-II;STRICOM;DIS/ADS;	12
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

ADVANCED DISTRIBUTED SIMULATION TECHNOLOGY II (ADST II)

DELIVERY ORDER # 0018
CDRL AB04

GENETIC ALGORITHMS AND EVOLUTIONARY PROGRAMMING

FOR: NAWCTSD/STRICOM
12350 Research Parkway
Orlando, FL 32826-3224
N61339-96-D-0002
DI-MISC-80711

BY: Lockheed Martin Corporation
Martin Marietta Technologies, Inc.
Information Systems Company
12506 Lake Underhill Road
Orlando, FL 32825

DOCUMENT CONTROL INFORMATION

Revision	Revision History	Date
	Original release	7/15/96
A	Incorporated customer comments	1/10/97
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		
-		

Table of Contents

PREFACE

This research activity examines the current state-of-the-art in modeling the command decision process and implementing such models in software. The primary and initial target application is in automated command agents for DIS/ADS. This report was prepared for the Command Decision Modeling ADST II Delivery Order in accordance with the following documents:

a) Command Decision Modeling Overview (ADST-II-CDRL-023A-9600236)
b) Functional Description of a Command Agent (ADST-II-CDRL-023A-9600237)
c) Rule Based Systems (ADST-II-CDRL-023A-9600238)
d) Genetic Algorithms and Evolutionary Programming (ADST-II-CDRL-023A-9600239)
e) Petri Nets and Colored Petri Nets (ADST-II-CDRL-023A-9600240)
f) Neural Networks and Bounded Neural Networks (ADST-II-CDRL-023A-9600241)
g) Case-Based Reasoning (ADST-II-CDRL-023A-9600242)

1.0 OVERVIEW OF TECHNOLOGY AREA

1.1 BACKGROUND

Genetic algorithms (GAs) have been around for a number of years. Interest in genetic algorithms began in the 1970's when John Holland introduced what is referred to as his Simple Genetic Algorithm (SGA) and gave a complete architecture and implementation technique [Srinivas & Patnaik 1994]. In the last few years, they have begun to gain popularity as effective optimization and search methods for large, difficult search problems where the search space is non-convex. In general, non-convex problems are known to be intractable with only those problems with few variables being solvable by traditional optimization methods such as gradient descent [Vavasis 1995].

Evolutionary programming (EP) is closely related to genetic algorithms. Both are based on the survival-of-the-fittest principle accredited to Darwinian theories of evolution and natural selection. These techniques start with an initial set of data and then iteratively change the data in order to approach solutions to a particular problem. Therefore, both GAs and EP have proven to be robust optimization methods. The difference being that GAs rely heavily on *crossover* while evolutionary programming emphasize *mutation*. Crossover and mutation are functional operators that are performed on a *population* of data sets. These operators get their names from science of genetics. In reproduction, chromosomes routinely exchange genetic code as part of a natural process. This genetic code comes from the male and female strings of chromosomes. The exchanging of genetic code is called crossover. The resulting child is a new data set which enters the population. In GAs, the selection of "chromosomes" for crossover is carefully controlled so that promising strings are selected to have more offspring than other strings. Mutation is also a somewhat frequent occurrence in nature. Mutation is used in GAs because crossover techniques using better strings to influence a population will eventually create a population where a majority of strings contain much the same genetic make-up. Therefore, new radical looking strings become impossible and the population could converge around a local optima. Mutation makes it possible for radical strings to be introduced and a global optima to be reachable.

A population is simply a finite set of solutions in the search space where an infinite number of solutions may be possible. These techniques are intended to find the optimal solution in this search space by iteratively modifying the population by increasing the number of promising solutions in the population and removing or decreasing the number of less promising solutions. A promising solution is decided according to a fitness function which is used to rank the solutions. A fitness function is simply the normalized version of the objective function which is the representation of the problem to be solved. In GAs, each solution may be likened to a string of chromosome in that it is made up genetic code which is shared or changed to make new solutions that enter the new

population. In essence, GAs search for the sequence of genetic code that provides the globally optimal solution to the objective function. Evolutionary programming also searches for a globally optimal solution. However, it does so based on the behavioral traits passed down from parent to child, *phenotype*, rather than the genetic code, *genotype* [Fogel 1994]. Genotypes represent the blueprint of an individual. Phenotypes represent how the items of the blueprint are manifested in the individual.

A notable technique that also mimics physical processes to perform heuristic search is called simulated annealing. This technique models the methods used to strengthen metals by leveraging the thermodynamic properties of materials. The metal is heated and cooled in a particular way to achieve the optimal alignment of atoms in the material. In contrast to GAs, simulated annealing is performed on one string which is replaced by a new string if the new string is better. Since, there is only one string, it is not possible to apply the crossover operator. It is possible for a worse string to replace a better string. The probability of this is based on the current temperature of the process. Higher probabilities for higher temperatures. The temperature starts high and then the process undergoes a cooling schedule. The temperature is lowered when the best string has been modified a number of times without finding an improved string. The process is terminated when the temperature is reduced a number of times without finding a better string.

1.2 ANATOMY OF GENETIC ALGORITHMS

The basic algorithm for genetic algorithms is illustrated below.

```
Initialize population (arbitrarily or otherwise)
Calculate fitness values of population members
LOOP While (Termination Criteria is Not Reached)
        Apply selection criteria
        Generate offspring
        Apply crossover operator
        Apply mutation operator
        Generation = Generation + 1
        Calculate fitness values of population members
END LOOP
```

1.2.1 Encoding Mechanism

Before this algorithm can be applied, a proper encoding mechanism for the strings must be selected. Traditionally, according to Holland's SGA, the GA is best optimized by a binary encoding scheme which imparts a characteristic known as implicit parallelism [Fogel 1994]. The idea is that based on the fitness value for a particular schema of binary code, one can derive information about the fitness value of strings matching that schema. A binary encoding maximizes the number of strings in the search space that would match

the schema. However, this has been found not to be the case for all optimization problems and is certainly a candidate for further study.

If one were to use the binary encoding method, a simple mapping from variable values to manageable integers would suffice to create the needed concatenated string of binary numbers.

1.2.2 Selection and Offspring

During selection, the survival-of-the-fittest principle is applied. Each string is assessed a fitness value according to the fitness function. Strings with better fitness values are given a higher probability to produce offspring than those with lower fitness values. Therefore, the strings with stronger gene sequences will pass them on to serve as building blocks towards an optimal solution. The proportionate selection scheme is a popular method for generating offspring. Using this method, the number of offspring for a particular string is determined by its fitness value divided by the average of all the fitness values of the population. The resulting number may be rounded as needed.

It has been shown that Holland's SGA without modification will not find a global optima [Gunter 1994]. Therefore, minor modifications to its mutation and selection strategies are needed to make convergence possible. The adoption of an elitist strategy is a sufficient change. An elitist strategy simply retains the best solution in the population from generation to generation. In this way, optimal solutions are not destroyed by crossover.

1.2.3 Crossover

Crossover is performed on two randomly selected strings based on a predetermined probability referred to as the crossover rate. The three main techniques for crossover are single-point, two-point, and uniform. In single-point crossover, the strings are broken at a randomly determined location and then the remainder of each string is switched with the other to create two new strings. This introduces biases since bits at the ends of strings tend to change more frequently than bits towards the middle of the strings. Two-point crossover is used to remove this bias. Strings are segmented using two randomly chosen locations and the segments are then switched between the two strings. Uniform crossover dictates that each bit location in the pair of strings has an equal probability of being switched. The drawback of this method is that it tends to break up the building blocks of genetic sequences that strive toward optimal solutions. Since whole segments are moved in the one-point and two-point methods, these building blocks are preserved.

1.2.4 Mutation

Each bit of every string is subjected to a probability of mutation. This probability is called the mutation rate. In GAs, the mutation is usually very small in comparison to crossover

rates. In the case of binary in coded strings, the mutation of a particular bit means to simply change it from a 0 to a 1 or vice-versa.

1.2.5 Control Parameters & Termination Condition

The control parameters of GAs are the population size, crossover rate, and mutation rate. Other operators besides crossover and mutation may also be applied to the strings and, therefore, would also be a source of control. Typical population sizes range up to 200 while crossover and mutation rates range from .5 to .95 and .001 to .05 respectively. Termination conditions are based on reaching a specified number of generations, on attaining a string with a desired high fitness value, or on achieving a considerable amount of homogeneity among the strings in the population.

2.0 NOTABLE IMPLEMENTATIONS AND VARIANTS

The popularity of evolutionary programming and especially genetic algorithms have increased in recent years. Much research has been done in many areas including operations research, robotics, and Very Large Scale Integrated (VLSI) circuit layout.

Because GAs provide good optimization techniques, the have been widely applied in operations research endeavors. NP-hard problems such as the traveling salesman problem (find the minimum distance for traveling between a number of cities) are favorably solved using GAs. These operations research problems provide practical use for resource constrained situations common to military logistical elements.

In the robotics industry, GAs are typically used for path planning and pattern recognition. Where efficiency and precision is a premium, GAs select the optimum balance of manipulator movement and torque reduction. Likewise, efficient routing of chip circuitry in VLSI designs are important in reducing cost.

3.0 TECHNICAL DETAILS

Genetic algorithms and evolutionary programming are iterative techniques for optimization and search. As such, these techniques may be very slow to converge to global optima and may not even attain an optimal solution. However, they are good at solving non-convex optimization problems that are generally intractable. The good news is that these techniques may be implemented in a parallel fashion.

4.0 APPLICABILITY TO COMMAND DECISION MODELING

4.1 POTENTIAL AS A PRIMARY OR SOLE TECHNICAL APPROACH

Genetic Algorithms and Evolutionary Programming rely on optimization of objective functions. Complex objective functions with many variables may result in a non-convex search space and non-convergence. The potential complexity of command decision models and the near real-time, reactive nature of command agents make the use of GAs and EPs very unlikely integrators of command agent architectures.

4.2 SUBPROBLEMS ESPECIALLY SUITED TO THIS TECHNOLOGY

As shown in current work with robotic and VLSI systems, GAs can be used to perform path planning activities that would be of use to a command agent. Pellazar (1994) uses GAs for vehicle route planning. Other planning activities such as operational and tactical planning should also be considered. As an off-line strategy for developing operational and tactical techniques, evolutionary algorithms could be useful in maximizing effectiveness while minimizing time. An example of this would be to consider a mission to be a solution to an objective function that characterizes the effectiveness of winning a battle or resupplying a unit. The possible tasks in the mission may be mapped to a binary encoded string and evolved using a GA. New solutions may be evaluated based on the sequence of tasks executed via a simulated unit. In this manner, new doctrine may be developed for the unit of the 21 century.

Recent work by Adamson and Joshi (1996) used the Close Action Environment (CAEN) simulation to study behavior changes in posture, speed, sensors, target acquisition, and types of fire. The fitness function used for a string of activities carried out in this simulation is based on the number of friendly forces killed and the time taken to reach the objective. Iteration of the GA continues until a stable sequence of activities is achieved. The researchers used a fairly restricted number of variables and tasks but illustrated interesting and successful use of GAs in the development of tactical behavior. Another example of using GAs to develop combat behavior can be found in [Smith 1995].

Fogel et. al. (1996) also presents a similar application using evolutionary programming. This innovative approach dynamically modifies mission plans in ModSAF to adapt to a changing environment. This is accomplished by mutating a population of mission plans for units and evaluating the population against an objective function that is based on a number of probabilities such as the probability of a kill between two vehicles. Once an optimized population is found, these mission plans are reassigned to the units. This sequence is followed throughout one exercise. One can certainly argue about how well these probabilities approximate the battlefield situation and whether effective optimization can be achieved in this manner but the authors state that results show successful mission adaptation on the simulated battlefield. This technique could be

modified evaluate exercise results and therefore evolve optimized behaviors as in [Adamson 1996]. However, the execution of that method is sure to be time consuming.

4.3 WEAKNESSES AND CONCERNS

As mentioned, the solutions to optimization problems provided by these evolutionary techniques may not converge to a global optimum. Can the command agent make use of sub-optimal solutions?

5.0 CONCLUSION

Genetic algorithms and evolutionary programs are not viable options for the representation of command agents. However, these techniques may be used in a near-real-time supporting role for the development and analysis of new and existing command and control procedures, the optimization of logistical functions, and the planning of unit routes.

6.0 REFERENCES

Adamson J. and Joshi, K.G. "Genetic Algorithms and Force Simulation." In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation.* Orlando, FL. July 23-25, 1996. pp. 237-242.

Andriole, Stephen J. "Information Technology for Command and Control." *IEEE Transactions on Systems, Man, and Cybernetics.* Vol. 16, No. 6. November/December, 1986. pp. 762-764.

Fogel, David B. "A Comparison of Evolutionary Programming and Genetic Algorithms on Selected Constrained Optimization Problems." *Simulation.* Vol. 62:4. June, 1995. pp. 397-403.

Fogel, David B. "An Introduction to Simulated Evolutionary Optimization." *IEEE Transactions on Neural Networks.* Vol. 5, No. 1. January, 1994. pp. 3-11.

Fogel, Lawrence J.; Porto, V. William; and Owen, Mark. "An Intelligently Interactive Non-Rule-Based Computer Generated Force." In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation.* Orlando, FL. July 23-25, 1996. pp. 265-271.

Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley. Reading, MA. 1989.

Gunter, Rudolph. "Convergence Analysis of Canonical Genetic Algorithms." *IEEE Transactions on Neural Networks.* Vol. 5, No. 1. January, 1994. pp. 96-101.

Holland, John H. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press. Ann Arbor. 1975.

Pellazar, Miles B. "Vehicle Route Planning with Constraints Using Genetic Algorithms." In *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON).* Vol. 1, 1994. pp. 111-118.

Smith, Robert E. and Dike, Bruce A. "Learning Novel Fighter Combat Maneuver Rules via Genetic Algorithms." *International Journal of Expert Systems.* Vol. 8, No. 3. 1995. pp. 247-276.

Srinivas, M. and Patnaik, Lalit M. "Genetic Algorithms: A Survey." *IEEE Computer.* June, 1994. pp. 17-26.

Vavasis, Steven A. "Complexity Issues in Global Optimization: A survey." *Handbook of Global Optimization*. R. Horst and P.M. Pardalos, Eds. Kluwer Academic Publishers. Netherlands. 1995. pp. 27-41.